Vehicle Routing Problem(VRP)

Goal

To optimize the delivery cost while ensuring that all delivery locations are visited and all demands are met. Choose one metaheuristic algorithm to solve the task.

Algorithm Choice: Genetic Algorithms(GAs)

GAs are inspired by natural selection and start with a randomly generated population of potential solutions. Each solution's fitness is evaluated to determine its effectiveness. The best solutions are selected as parents, undergo crossover to produce new offspring, and mutation to maintain diversity. This iterative process of evaluation, selection, crossover, and mutation continues for multiple generations until a termination condition, such as a set number of generations, is met, leading to progressively better solutions.

In hindsight, the VRP should be solved using Ant Colony Optimization due to its nature designed for routing problems. But in this case, GAs are preferred because they offer flexibility in handling complex constraints, the ability to explore multiple solutions simultaneously, and an effective balance of exploration and exploitation. Moreover, compared to other metaheuristic algorithms, GAs avoid local optima and are easier to implement and adapt, making them a robust and efficient choice.

Implementation Details

Functions:

```
def calculate_distance(point1, point2): Calculate Euclidean distance

def generate_initial_population(pop_size, customers, vehicles_data):

Generate the initial population solutions.
```

```
def evaluate(solution): Evaluates the fitness of each solution based on
total cost and capacity constraints.

def selection(population, fitnesses):

Selects the best solutions to act as parents for the next generation.

def crossover(parent1, parent2):

Performs crossover between two parent solutions to produce offspring.

def mutate(solution, mutation_rate=0.1):

Introduces mutations to maintain genetic diversity.

def genetic_algorithm(customers, vehicles_data, pop_size=100,
generations=500, mutation_rate=0.1):

The main genetic algorithm functions to find the best solution.

def format_output(best_solution):

Formats the best solution into the desired output.
```

Result:

With 10 customers as the data , at best the algorithm results in 3 vehicles solutions with a run time of 2.6s. After I added customers' data to 30, the algorithm resulted in 8 vehicle solutions with a run time of 8.7s.

Below is one of the sample output:

```
Total Distance = 155.938 km
Total Cost = RM 196.00

Vehicle 1 (Type A):
Round Trip Distance: 109.489 km, Cost: RM 131.39, Demand: 23
Depot -> C8 (11.612 km) -> C9 (11.358 km) -> C7 (42.734 km) -> C4 (25.115 km) -> C3 (8.323 km) -> Depot (10.347 km)
Vehicle 3 (Type A):
Round Trip Distance: 16.871 km, Cost: RM 20.25, Demand: 8
Depot -> C10 (8.436 km) -> Depot (8.436 km)
Vehicle 2 (Type B):
Round Trip Distance: 29.577 km, Cost: RM 44.37, Demand: 26
Depot -> C2 (9.072 km) -> C1 (5.012 km) -> C5 (4.839 km) -> C6 (2.594 km) -> Depot (8.061 km)
```